GESTION DE RESSOURCES, ORDONNANCEMENT, ORCHESTRATION

COMET SIL

31/05/2022

CLS

Guillaume Eynard-Bontemps

cnes

Sommaire

1

INTRODUCTION

Gestion de ressources vs Ordonnancement vs Orchestration de tâches

Gestion ou Orchestration de ressources

- **Affecter des tâches ou processus finis ou continus à un ensemble connu de ressources physiques (CPU, RAM, Disque, Réseau)**
- **Valable pour un serveur unique comme pour des clusters de milliers de nœuds.**

Ordonnancement/Scheduling

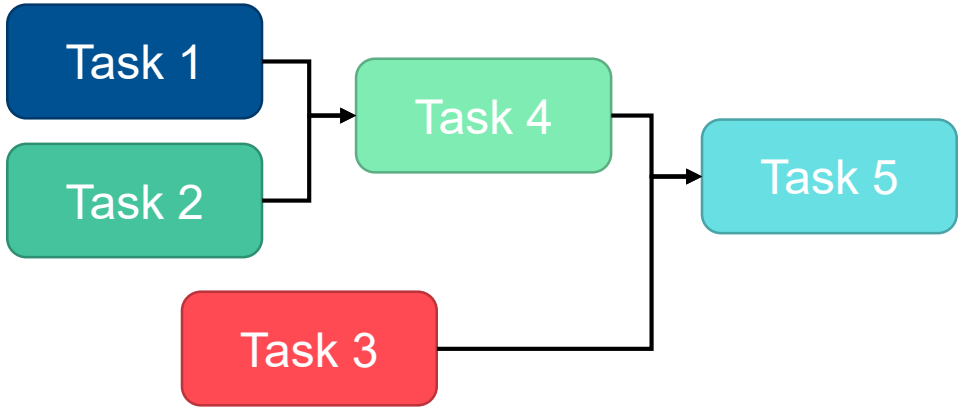
- **Exécuter les tâches sur des ressources dans un certain ordre dépendant de certaines caractéristiques :**
  - Ressources demandées,
  - Priorité,
  - Partage des ressources,
  - Durée de la tâche, etc.

Orchestration de tâches

- **Exécuter des tâches de longueur finies dans un ordre défini, en utilisant différents concepts :**
  - Dépendances entre tâches, Séquence ou workflow de tâches
  - Priorité,
  - Déclenchement sur évènement,

System Orchestration
Wikipedia (en) : ”In system administration, [illegible] configuration, coordination, and management of computer systems and software [illegible], Puppet, SALT, etc.)” https://en.wikipedia.org/wiki/Orchestration_(computing)

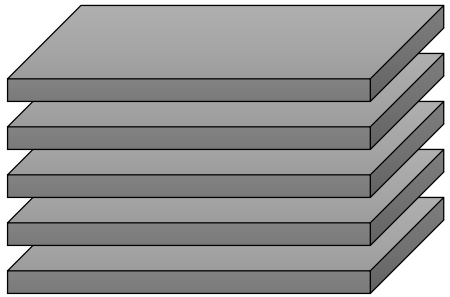
Orchestration repose sur la gestion des ressources et l'ordonnancement



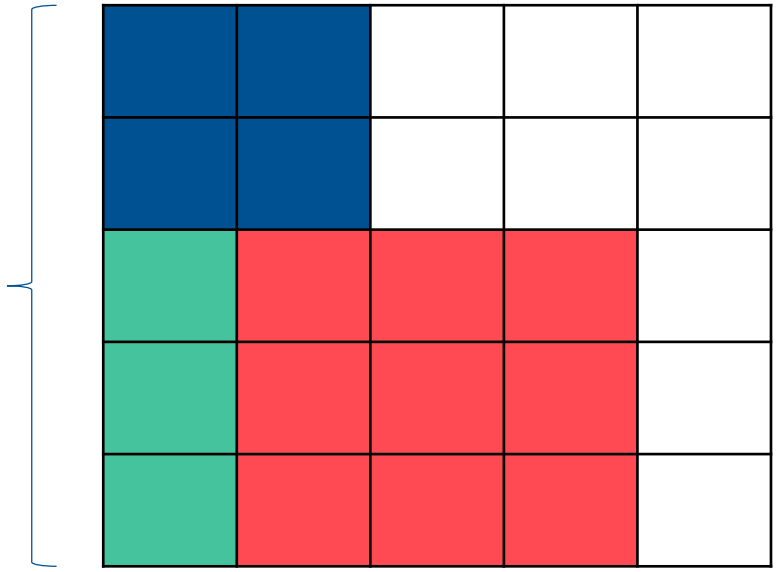
Orchestration de tâches (ou chorégraphie)

Ordonnancement

Gestion de ressources



Set of compute servers



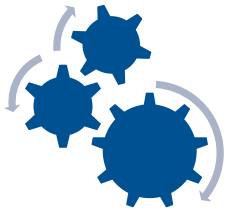
Schematic view of computing resources

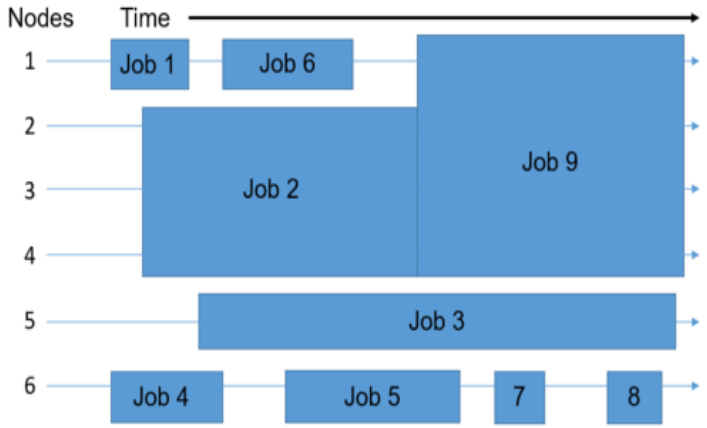
Ordonnancement vs Orchestration de tâches

Ordonnancement

- **Liste de tâches avec caractéristiques**
- **Ordonnancement à partir de ces caractéristiques : quelle tâche effectuer en premier.**
- **Ex de base : FIFO (ou alors pas d'ordonnancement ☺)**

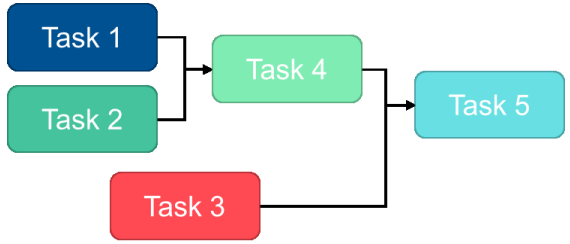
Job 1
Job 2
Job 3
Job 4
Job 5
Job 6
Job 7
Job 8
Job 9

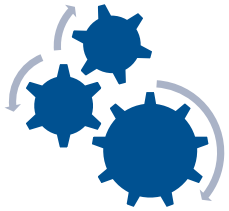




Orchestration de tâches

- **Liste de tâches avec dépendances**
- **Description centralisée d'une séquence de tâches : workflow ou pipeline**
- **Déclenchement automatique**
- **Repose sur l'ordonnancement et la gestion des ressources pour l'exécution.**





Job 1
Job 2
Job 3

Des frontières floues, et des situations et outils variés

Laptop or server

HPC Center

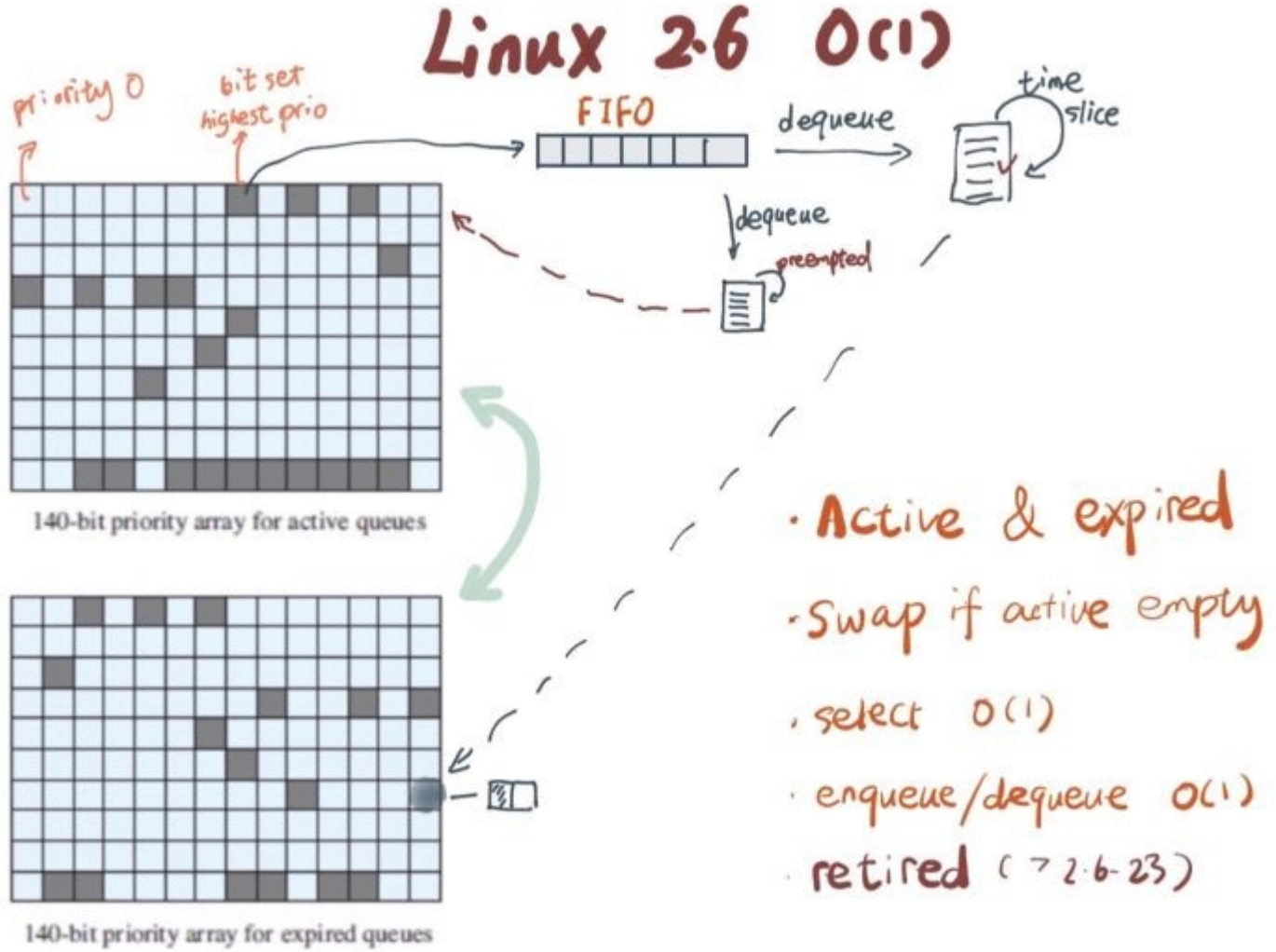
Cloud service

Several data centers

2

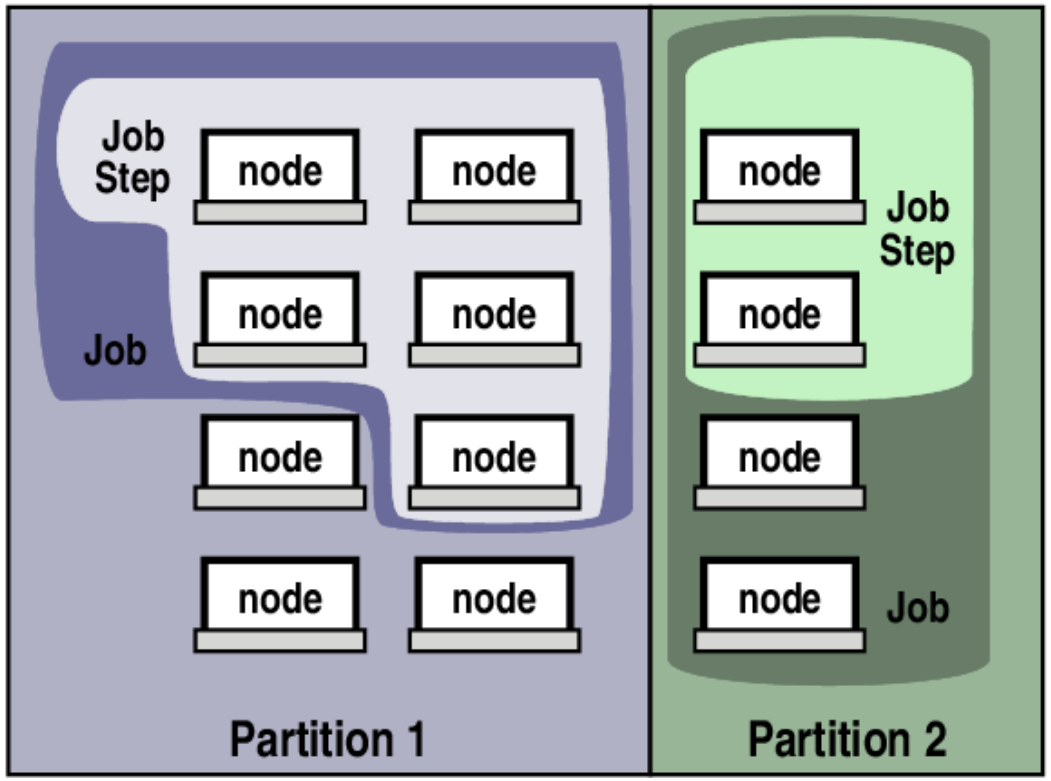
GESTION DE RESSOURCES

Le système d'exploitation : gestion et ordonnancement de ressources



High Performance Computing : gestion de ressources

- **Caractéristique d'un job**
  - Resources : CPU, RAM, GPU, Disque
  - Walltime : temps estimé,
  - Partition, ressources particulières,
  - Priorité
  - …



Gestion des ressources, par job

Cloud : Kubernetes, orchestrateur de conteneurs (et de ressources)

Pod/conteneur

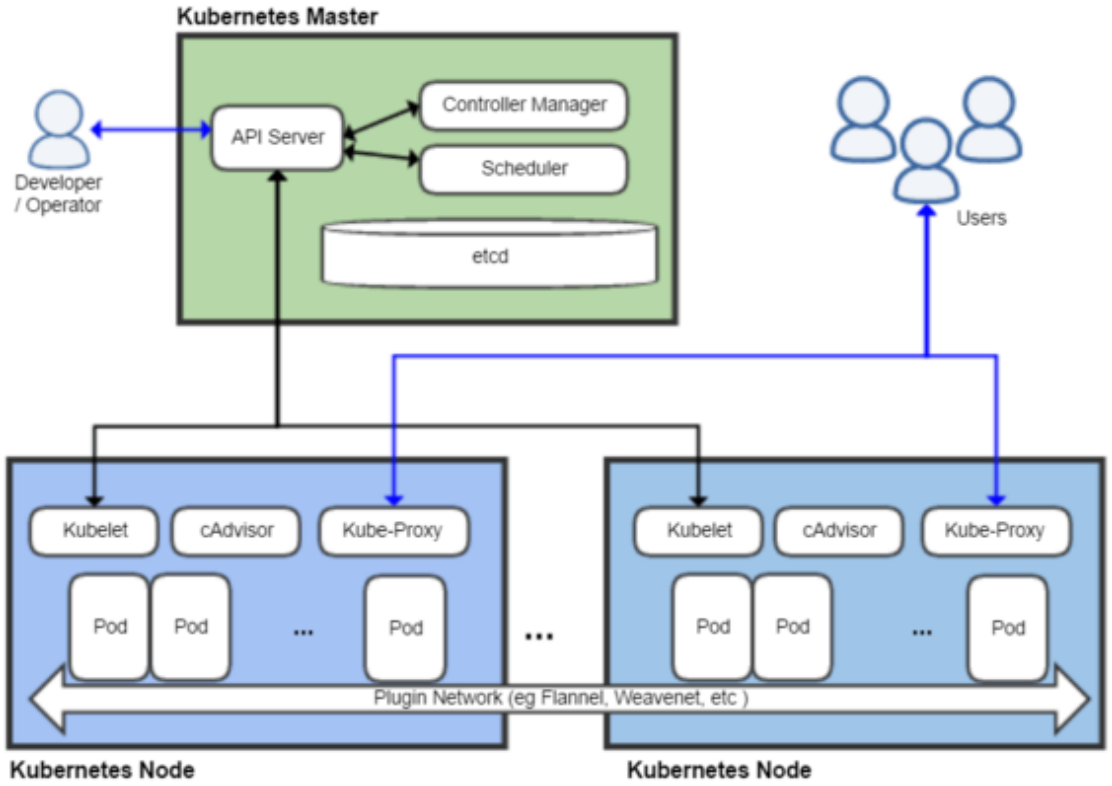
Pod/conteneur

Pod/conteneur

Pod/conteneur

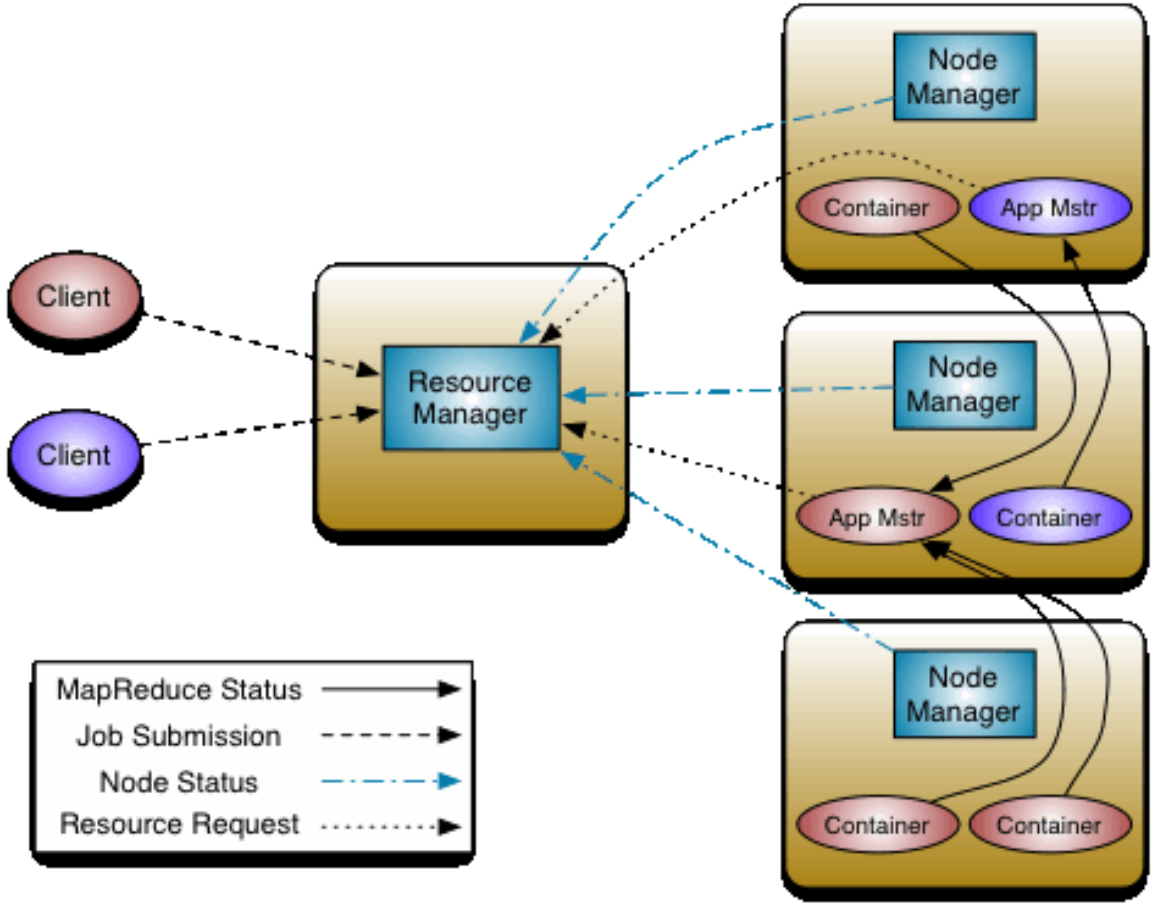
Helm chart ou spec

- **Caractéristique d'un Pod**
  - Resources : CPU, RAM, GPU, Disque
  - Partition, ressources particulières, Taints
  - Priorité
  - …

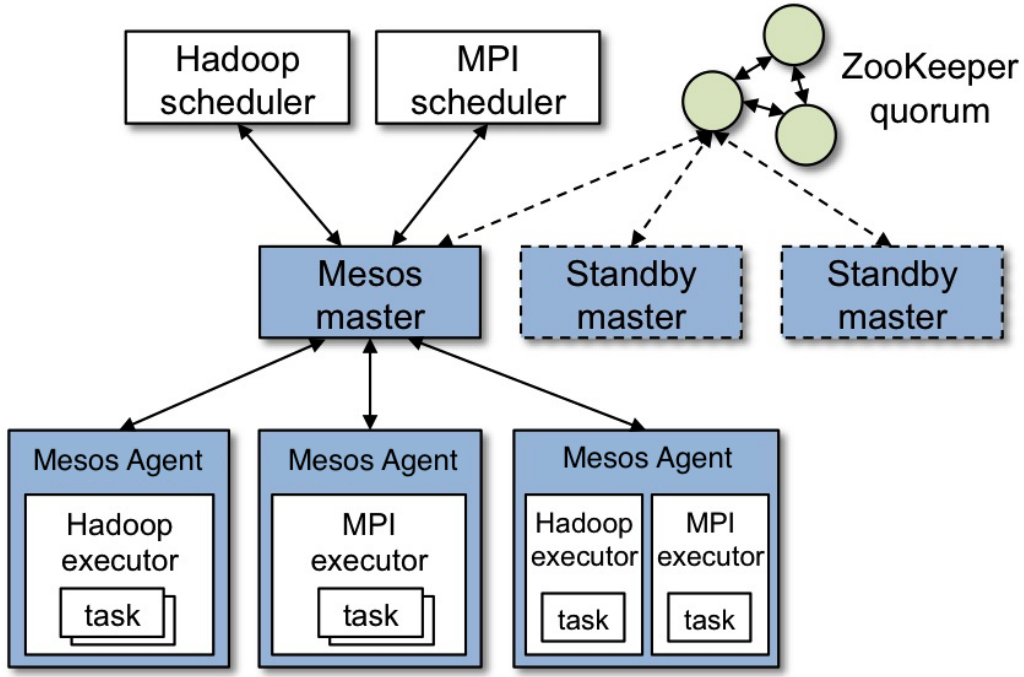


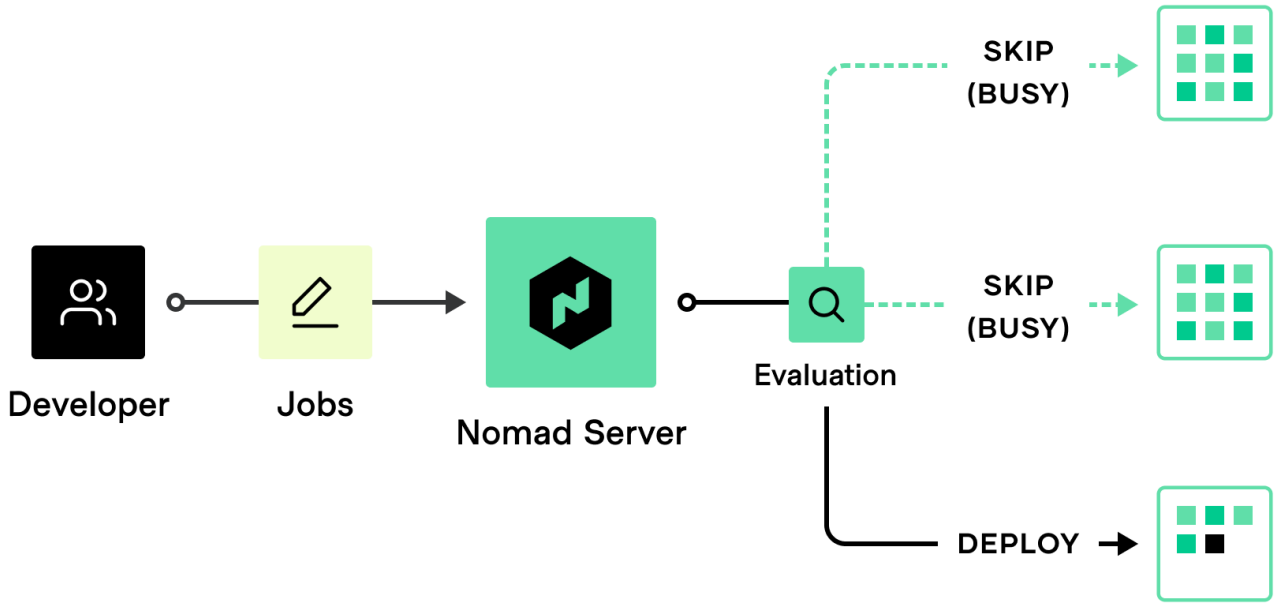
Ecosystème Hadoop : YaRN

- **Application/Conteneur (Spark, MapReduce)**
  - Sous-étapes
  - Resources pour chaque étape
  - Ressources du master
  - …



Et plein d'autres





Bilan gestion de ressources

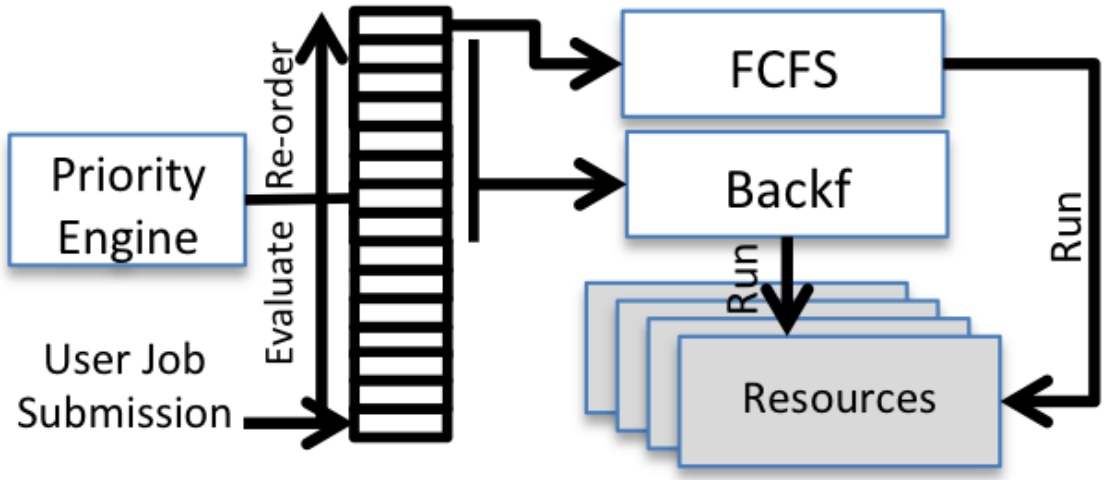
- **Architecture souvent identique : Client → Master → Workers**
- **On soumet des éléments bien définis : Job, Pods, Tasks, Applications, avec des caractéristiques**
  - CPU, RAM, Disk
  - Priorité
  - Durée
  - Teinte ou partition
  - Etc…
- **Toujours une part d'Ordonnancement ou Scheduling, plus ou moins complexe, pour affecter les process aux ressources de manière intelligente.**

3

ORDONNANCEMENT

Ordonnancement HPC

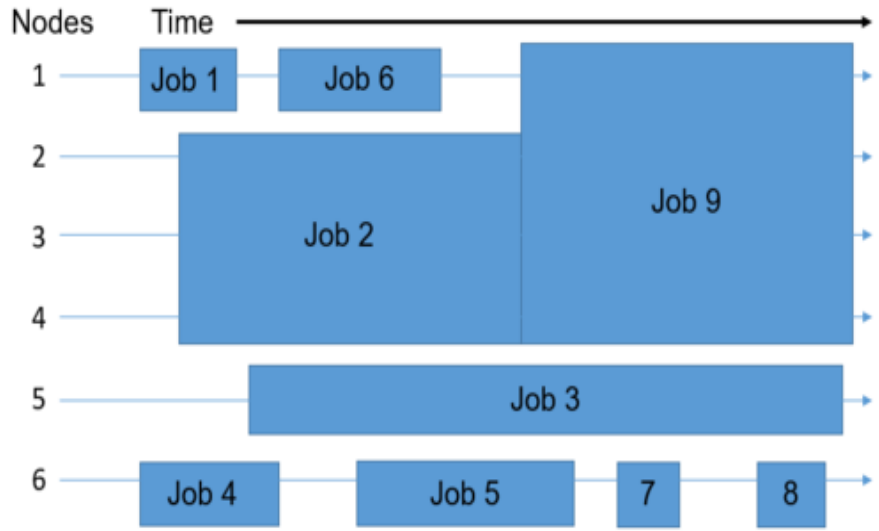
Job :
- Chunks,CPU,Mem,Disk
- Walltime
- Queue/Partition/QoS
- Project/Account
- Constraints/Options



PBS Works™

IBM Spectrum LSF

- **Ordonnance des jobs à travers la gestion d'une queue globale**
- **Proposent des partitions ou QoS (groupes de nœuds, priorités, limites particulières)**
- **Ordonnancement = remplir le maximum de ressources de calcul en satisfaisant les besoin utilisateurs**
  - Gestion de priorité des jobs : fonction des partitions ou QoS, **fairshare**, préemption, age du job (starvation), taille du job, etc.
  - Backfilling = remplir les trous une fois la priorité évaluée (besoin du walltime)
  - Préemption des jobs, arrêt des tâches moins prioritaires.



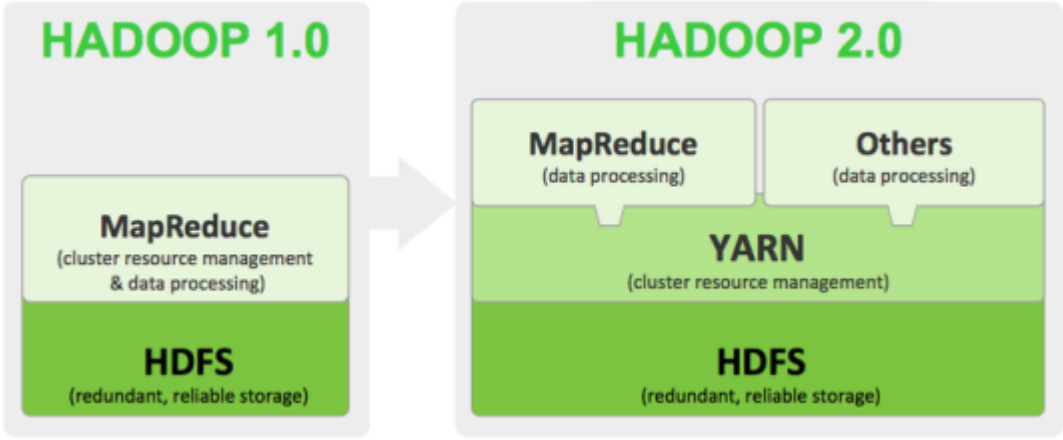
Ordonnancement HPC : orchestration ?

- **Propose souvent un mécanisme de dépendance : orchestration de jobs !**
  - Permet des workflows simples, exemple avec Slurm (mais existe dans PBS aussi)

```
#! /bin/bash

# first job - no dependencies
jid1=$(sbatch  --mem=12g --cpus-per-task=4 job1.sh)

# multiple jobs can depend on a single job
jid2=$(sbatch  --dependency=afterany:$jid1 --mem=20g job2.sh)
jid3=$(sbatch  --dependency=afterany:$jid1 --mem=20g job3.sh)

# a single job can depend on multiple jobs
jid4=$(sbatch  --dependency=afterany:$jid2:$jid3 job4.sh)

# swarm can use dependencies
jid5=$(swarm --dependency=afterany:$jid4 -t 4 -g 4 -f job5.sh)

# a single job can depend on an array job
# it will start executing when all arrayjobs have finished
jid6=$(sbatch --dependency=afterany:$jid5 job6.sh)

# a single job can depend on all jobs by the same user with the same name
jid7=$(sbatch --dependency=afterany:$jid6 --job-name=dtest job7.sh)
jid8=$(sbatch --dependency=afterany:$jid6 --job-name=dtest job8.sh)
sbatch --dependency=singleton --job-name=dtest job9.sh


# show dependencies in squeue output:
squeue -u $USER -o "%.8A %.4C %.10m %.20E"
```

Ordonnancement Big Data / Hadoop

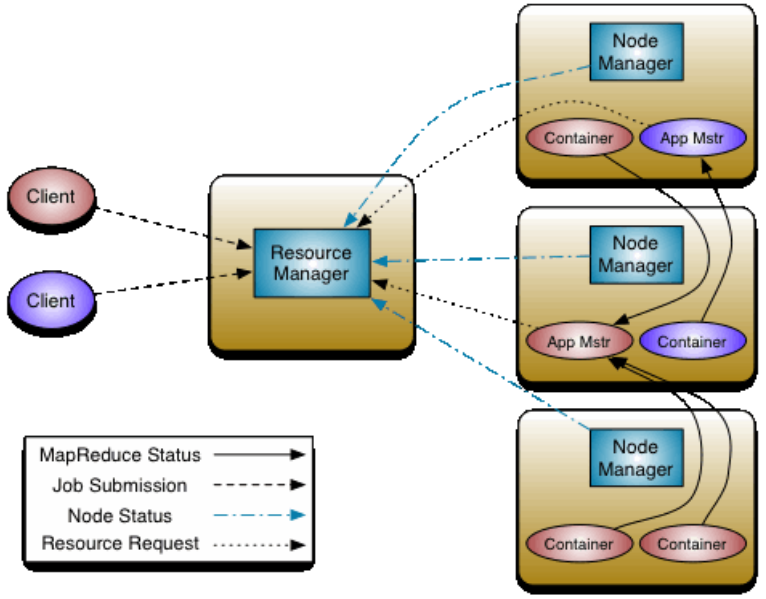
YaRN



Application/Container :

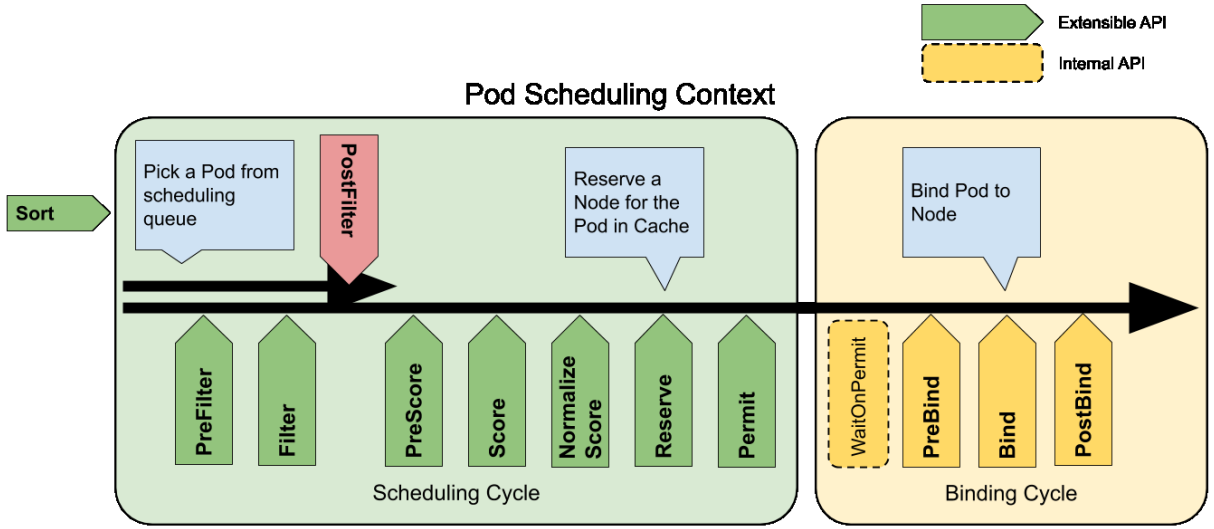
- Memory/CPU/Disk/Network
- Partition

- **Yet another Resource Negociator**
- **Gestion de ressource essentiellement**
- **Mais aussi Ordonnancement (ou Scheduling) : priorité, fairshare, capacité.**
- **Le reste est porté par l'ApplicationMaster**

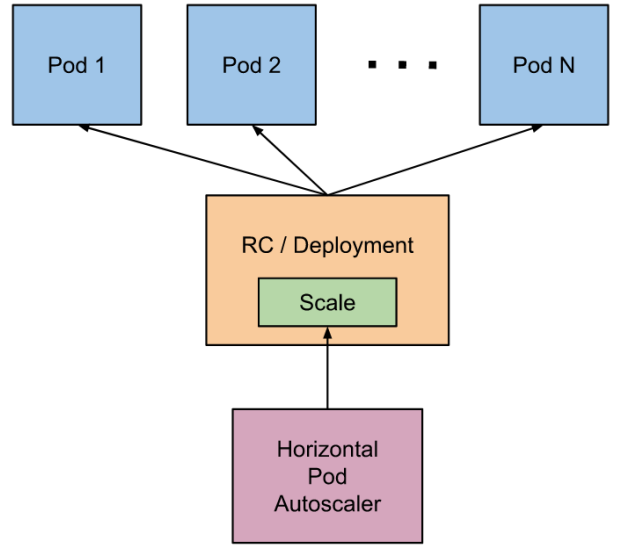


Ordonnancement Kubernetes

Pod :
- Resources requirements
- HW/SW/policy constraints
- Affinity
- Data locality
- Etc.



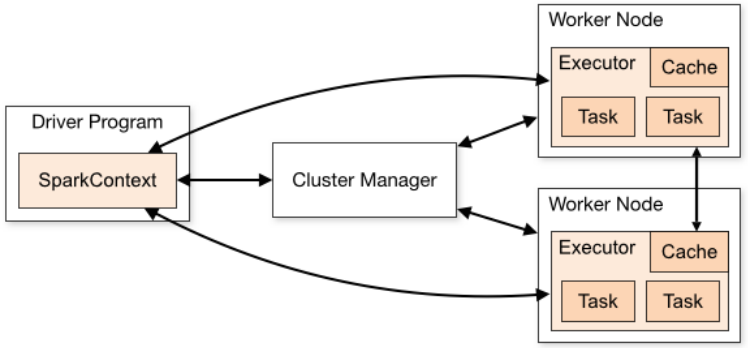
- **Kube-scheduler : Trouver le meilleurs nœud pour exécuter un POD**
- **POD a priori de durée infinie**
- **Résilience : s'assurer qu'un pod est dans l'état désiré quelque part.**
- **En 2 étapes :**
  - Filtrage
  - Notation
- **Scheduling Policies et Scheduling Profiles.**
- **Permet l'auto-scaling (Horizontal Pod Autoscaling)**

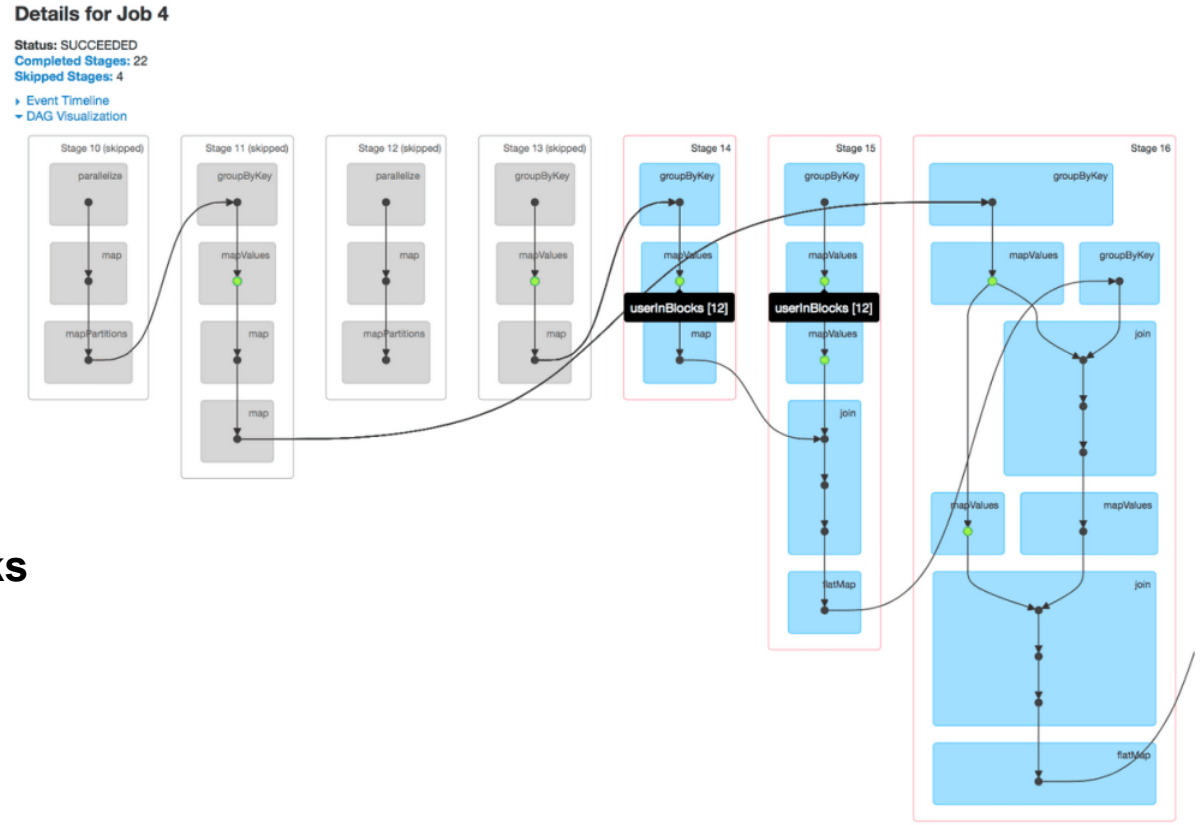


4

DAGS ET TRAITEMENTS DE DONNÉES DISTRIBUÉS

Big Data / Spark

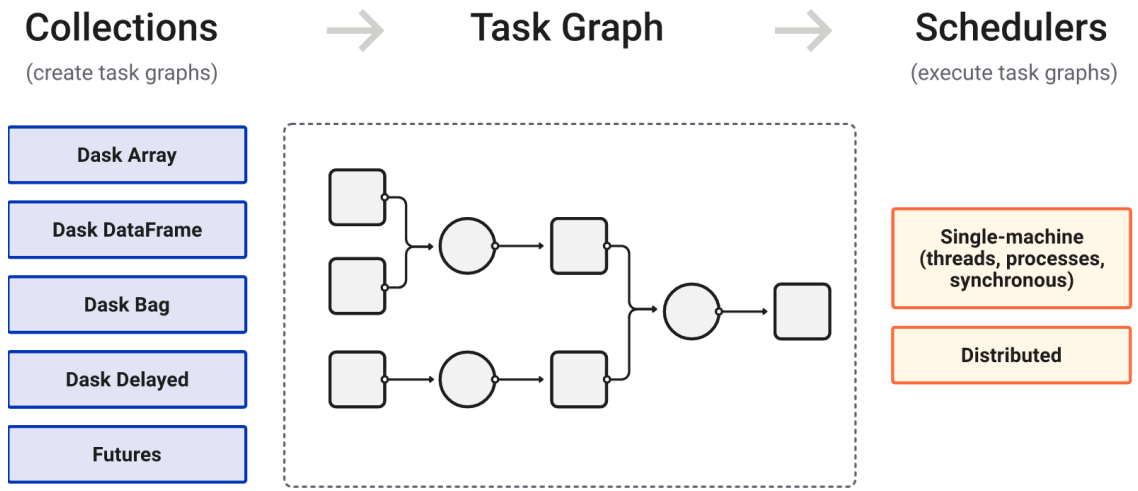




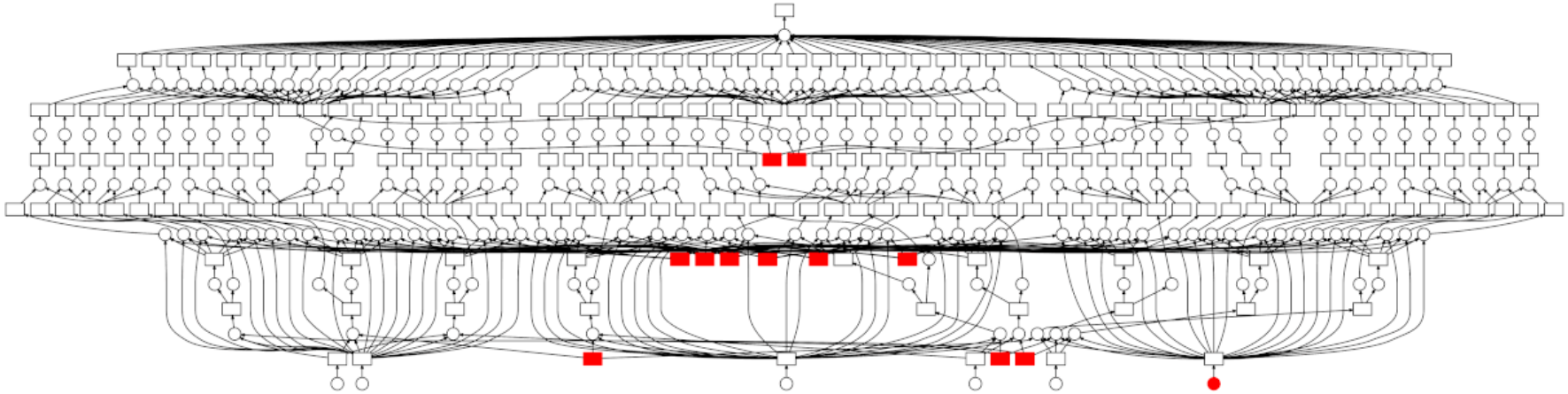
- **Application de <u>traitement de données distribuées</u>**
- **Exécute des jobs, découpés en stages, découpés en tasks**
  - Un job est un DAG (Direct Acyclic Graph)
  - Chaque étape peut lancer des milliers de tâches.
  - Découpage sur les données d'entrées
  - Job décrit par une suite d'opération sur les données.
- **Notion d'orchestration de tâches donc.**
- **Notion de gestion de ressources à travers les workers**
- **Mais gestion bas niveau généralement déléguée à YARN, Kubernetes ou Mesos (peut aussi s'exécuter en standalone).**

Focus sur Dask (1)

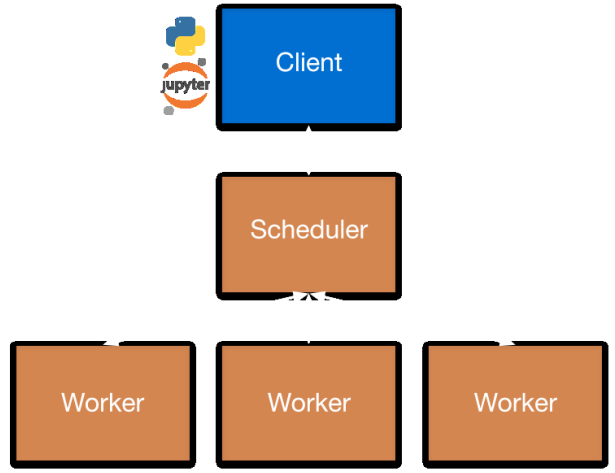
DAG :
- Liste de toutes les tâches
- Liste des dépendances
- Produit à partir des API de Dask



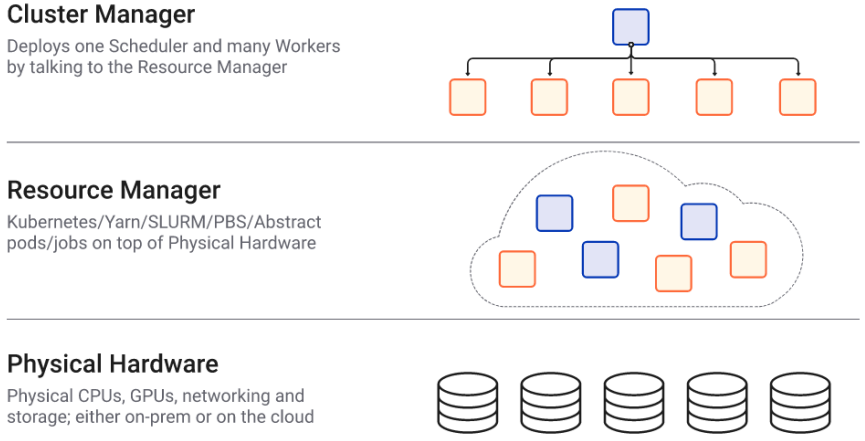
- **Traitements de données distribués, ou distribution de tâches arbitraires**
- **Tout est graphe (DAGs), composés de tâches fines**
- **Le scheduler est le maitre du graphe global, en assigne par défaut à des workers**



Focus sur Dask (2)



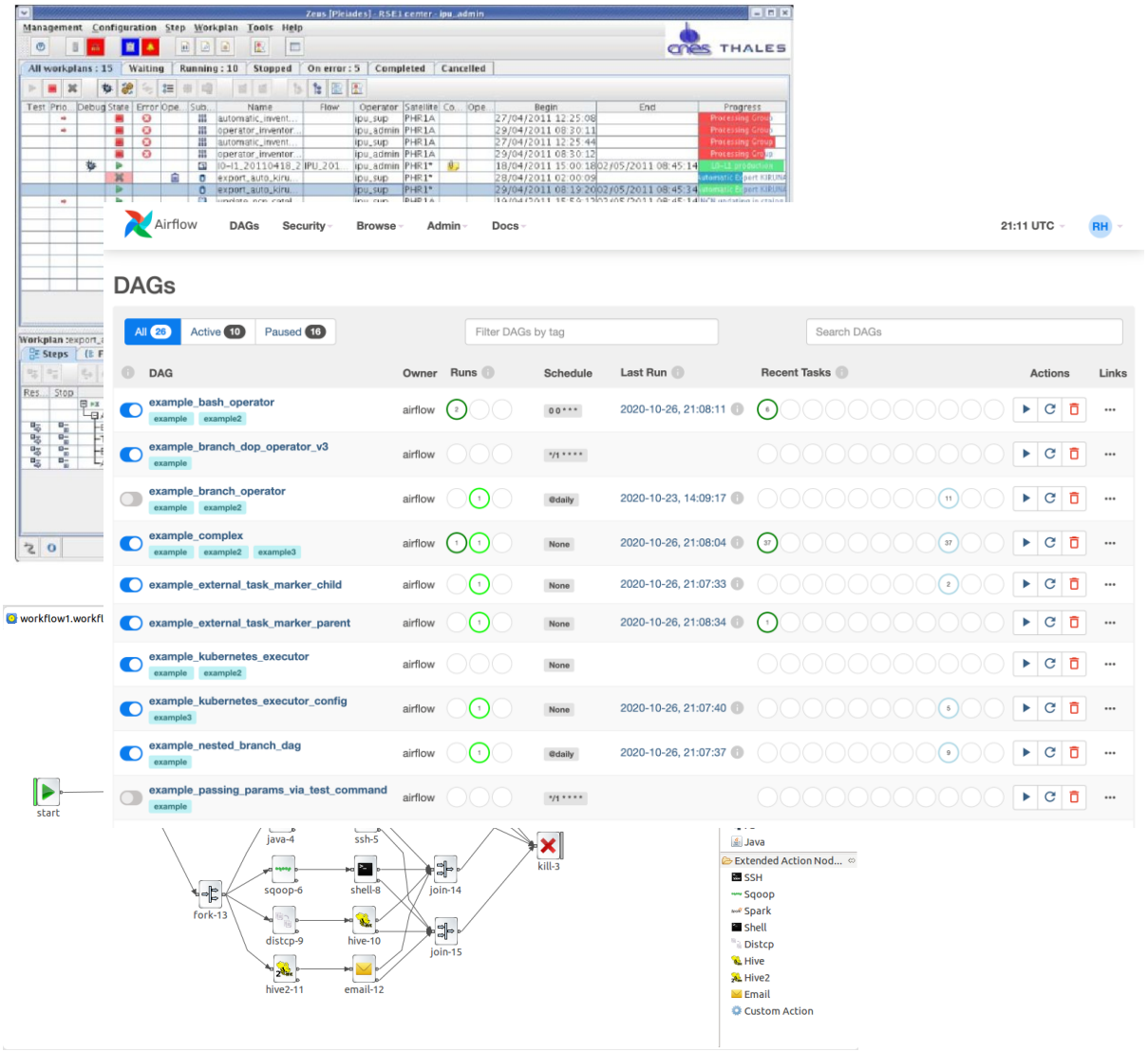
- **Scheduler : Master/Worker**
- **Optimisation complexe de l'ordonnancement des tâches pour limiter l'empreinte mémoire globale par exemple (depth first).**
- **Quelques fonctionnalités d'orchestration :**
  - task stealing (un peu de chorégraphie ☺)
  - Retry on error
  - Gestion des ressources en interne (ex GPU)
- **Attention : pas de méchanismes complexe de pause, réexécution, triggering**
- **Traitement a priori courts (qqs heures, qqs jours max). Tâches individuelles courtes : de qqs seconde à qqs minutes, 1 ou 2 h max.**
- **S'interface avec Kubernetes, YaRN, HPC, local (peut aussi s'exécuter en standalone).**



5

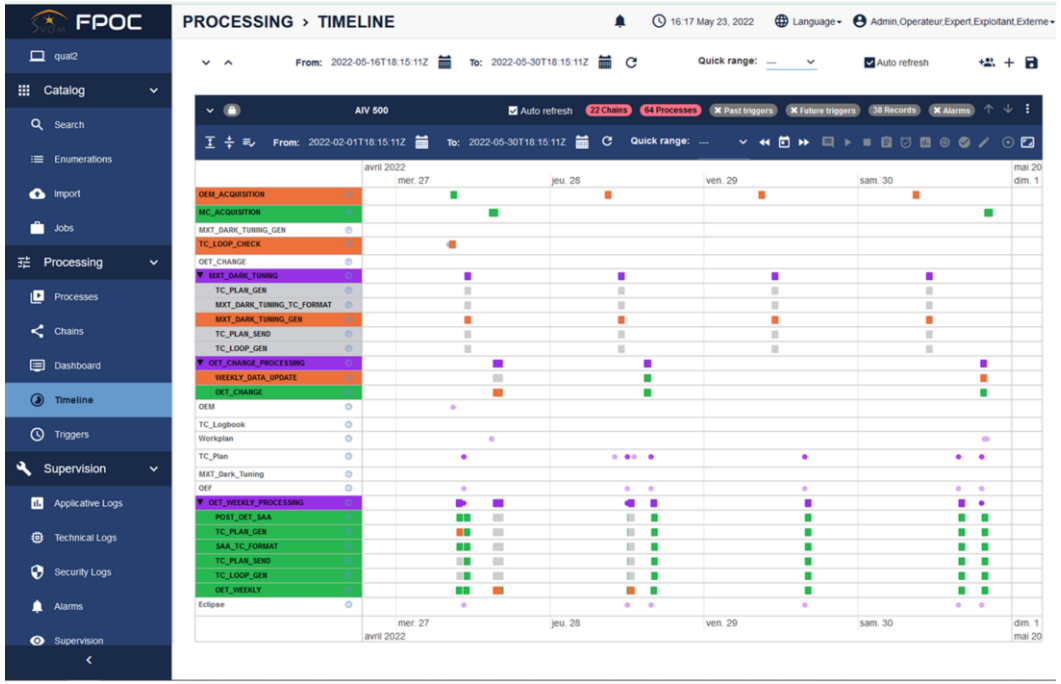
ORCHESTRATEURS DE TÂCHES OU WORKFLOWS

Orchestrateur de tâches / Workflows



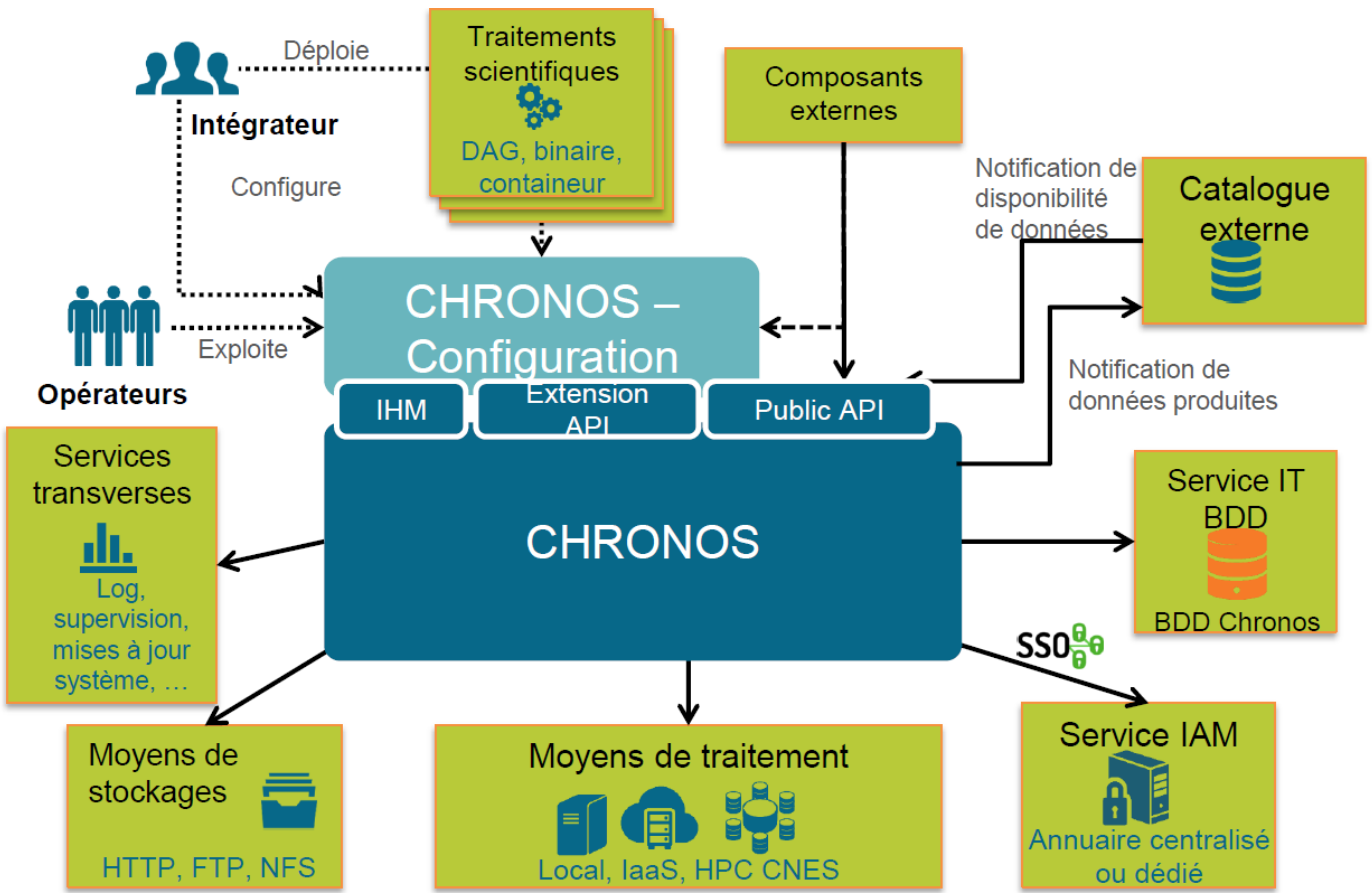
- **Au CNES : Phoebus, Spirit, Chronos, etc.**
- **Oozie pour Hadoop,**
- **Etat de l'art Data Science : Airflow.**
- **Fonctionnalités types**
  - Triggering de tâches (date, sur évènements)
  - GUI complète pour suivre les workflows (suite de tâches qu'on peut réexécuter).
  - Editeur de workflows/graphes/pipelines.
  - Possibilité de stop/restart d'un workflow. Retry/Error.
  - Workflow décrit par un fichier texte (syntaxe Xml, Yaml, Code, etc.), partageable et gérable en conf.
  - Exécution de tâches sur temps court (heures) ou long (jours)
  - Parralélisation/distribution des tâches
- **Orienté opération/production/automatisation.**
- **S'interfacent avec des gestionnaires de ressources (Kubernetes, HPC, Mesos, ou même Dask, Celery, surement d'autres).**

SPIRIT


FPOC
PROCESSING › TIMELINE
Catalog
Search
Enumerations
Import
Jobs
Processing
Processes
Chains
Dashboard
Timeline
Triggers
Supervision
Applicative Logs
Technical Logs
Security Logs
Alarms
Supervision


DATA
- Data Manage
- Catalog
PROCESSI
Creates, sched
and monitors all
proces
PostgreSQL
SPIRIT
MONITORING
- Alarms
- Logs
kibana
ACCESS
- Authentificati
- Users / Groups Management
IHM

Chronos - Architecture



TOIL
Massively Parallel Workflows
(and fire-breathing dragon slugs)

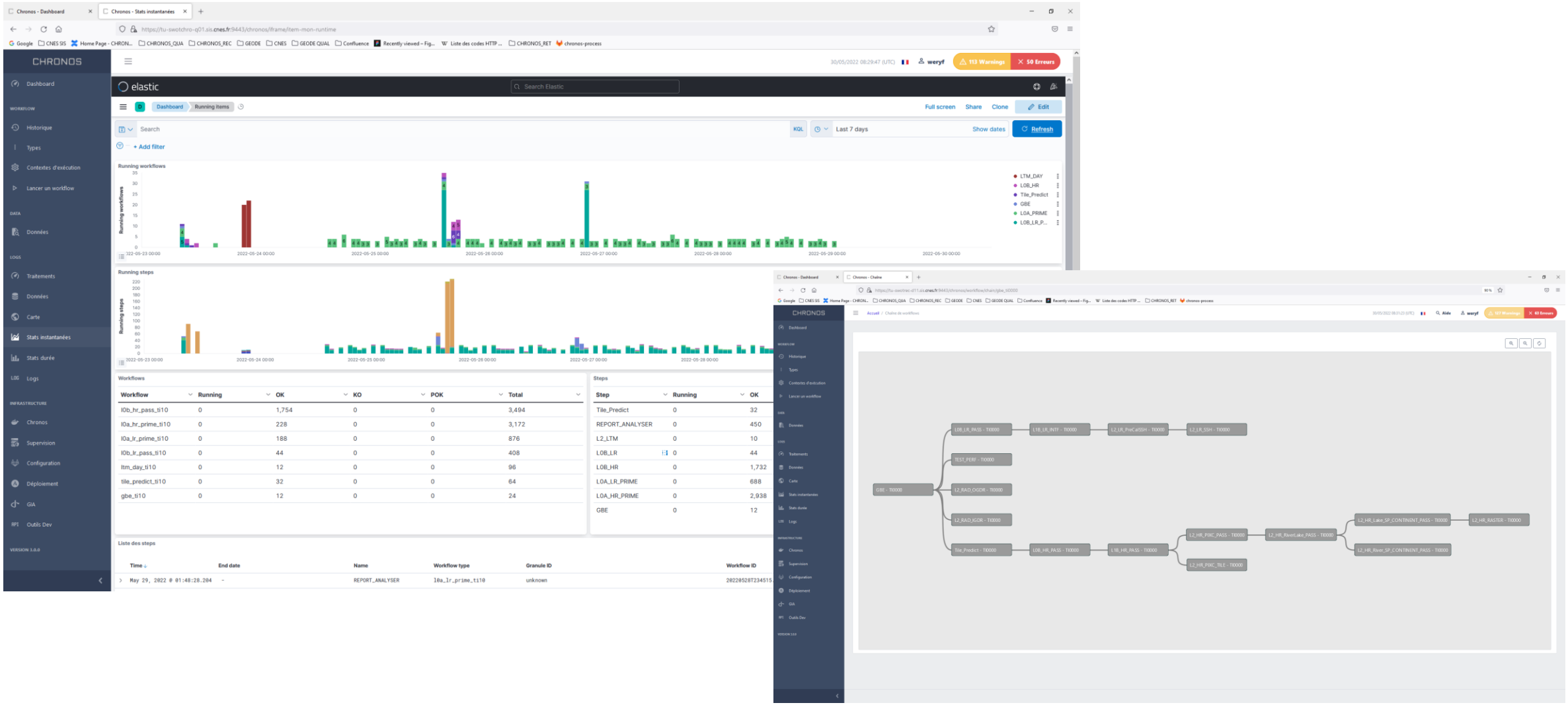
Apache MESOS

Designé pour le Cloud, optimisé pour HPC

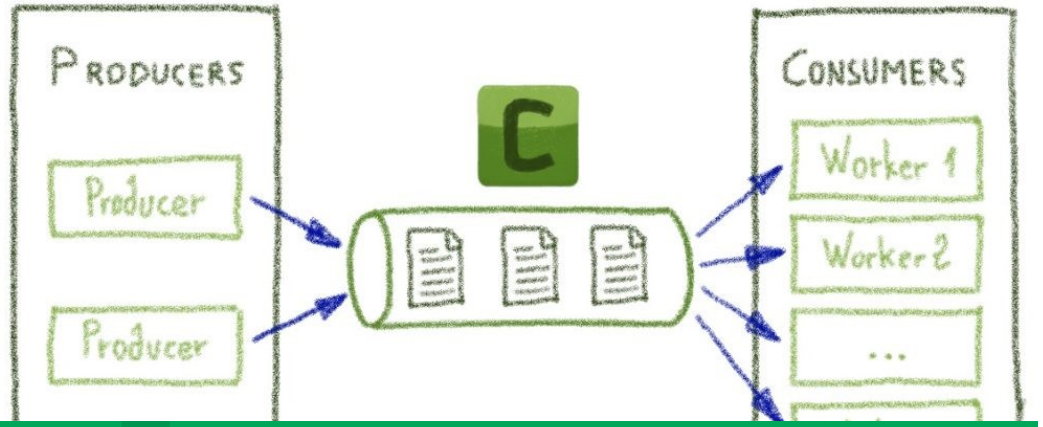
- Agnostique aux moyens de calculs,
- “Big Data for Big Data”

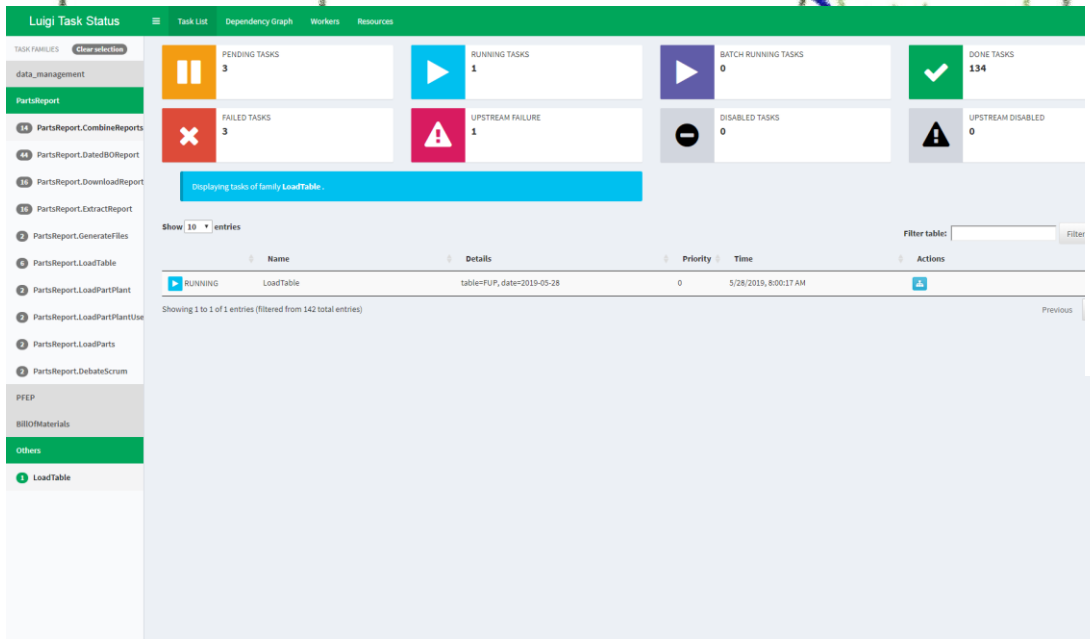
La solution a été utilisée pour traiter **20 000 échantillons d'ADN** (**108 To de données en entrée**) sur un cluster de **32 000 cœurs** sur Amazon AWS. Les **368 000 tâches** nécessaires aux traitements ont été réalisées en **90h**.

Chronos – IHM WEB



Exemples autres

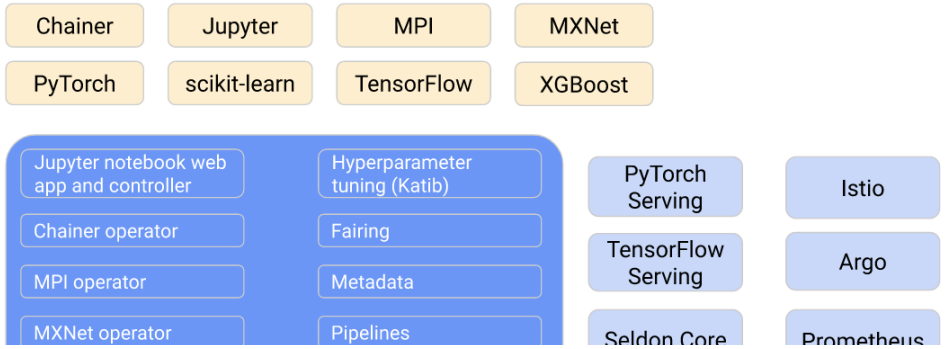


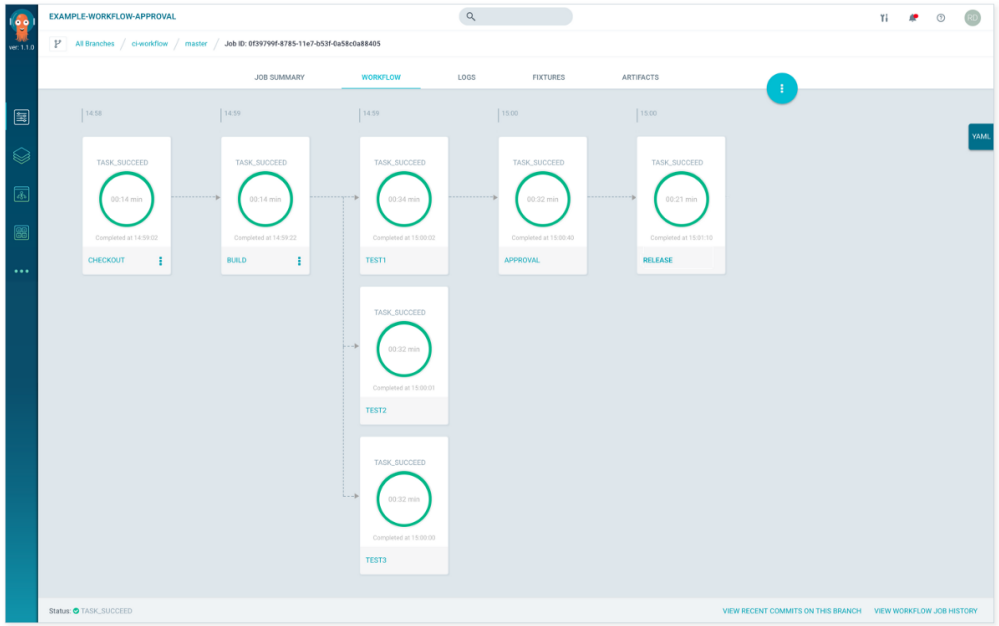


ML tools

Kubeflow applications and scaffolding

Platforms / clouds





6

CONCLUSION

Difficile de définir et d'avoir des frontières claires entre gestion de ressources, orchestration, ordonnancement ou chorégraphie.

- **La gestion des ressources repose sur l'ordonnancement,**
- **L'ordonnancement fait parfois de l'orchestration,**
- **L'orchestration de tâches propose souvent une gestion de ressources et de l'ordonnancement…**

L'outillage doit être sélectionné en fonction du besoin réel, il faut prendre le bon niveau.

- **Un ordonnanceur HPC peut être indiqué pour des workflows très simple,**
- **Dask pour du traitement de données portable sur plusieurs infrastructures,**
- **Un gestionnaire workflow plus haut niveau pour un centre de production.**

Exemples de caractéristiques clés pour choisir un ou plusieurs outils :

- **Type de job et caractéristiques des tâches à lancer**
- **Environnement et ressources disponibles (serveur, HPC, Cloud, multiple)**
- **Opération/Dev (Re-jeu des tâches en erreur, déclenchement automatique, etc.)**
- **Besoins/fonctionnalités : déclenchement auto, distribution massive, workflows complexes…**
- **Etc., à vous ?**

https://app.klaxoon.com/join/BJGTHKG